APPENDIX TO "Target Ligand Generation"

Naming Inventors: Albert Pierce and Guy Bemis

```
#!/usr/bin/python2.2

import sys,string,math
from openeye.oechem import *

worstDist = 1.0 #max distance b/w bonds across which molecules will be crossed
worstAng = 15.0 #max angle b/w bonds across which molecules will be crossed
worstDist = worstDist**2

def bondmatch(b1, b2):
#returns "matchsense" = 1 if Mol1.bond1.Begin corresponds to Mol2.bond2.Begin
#                     = 2 if Mol1.bond1.Begin corresponds to Mol2.bond2.End
#  matchsense = 0 if bonds aren't close(+/-worstDist) and parallel(+/-worstAng)
#             = 0 if bonds are of different order
  global worstDist

  parallel = 0
  antiparallel = 0
  if b1.GetOrder() != b2.GetOrder():
    return 0
  at11 = b1.GetBgn()
  at12 = b1.GetEnd()
  at21 = b2.GetBgn()
  at22 = b2.GetEnd()
  cart11 = mol1.GetCoords(at11)
  cart12 = mol1.GetCoords(at12)
  cart21 = mol2.GetCoords(at21)
  cart22 = mol2.GetCoords(at22)
```

```
34
35          dist1 = r2(cart11,cart21)
36          if dist1<worstDist:
37            dist2 = r2(cart12,cart22)
38            if dist2<worstDist:
39              if deadend(at11) and deadend(at21):
40                return 0
41              if deadend(at12) and deadend(at22):
42                return 0
43              parallel = 1
44
45          dist1 = r2(cart11,cart22)
46          if dist1<worstDist:
47            dist2 = r2(cart12,cart21)
48            if dist2<worstDist:
49              if deadend(at11) and deadend(at22):
50                return 0
51              if deadend(at12) and deadend(at21):
52                return 0
53              antiparallel = 1
54
55          if (parallel+antiparallel)==0:
56            return 0
57
58          ang = calcAngl(cart11,cart12,cart21,cart22)
59          if ang > worstAng:
60            return 0
61          if parallel==1:
62            return 1
63          else:
64            return 2
65
66        def deadend(atom):
67          count = 0
68          for nbor in atom.GetAtoms():
69            count += 1
70          if count < 2:
71            return 1
```

```
72            return 0
73
74          def r2(xyz1, xyz2):
75            dist2 = (xyz1[0] - xyz2[0])**2
76            dist2 = dist2 + (xyz1[1] - xyz2[1])**2
77            dist2 = dist2 + (xyz1[2] - xyz2[2])**2
78            return dist2
79
80          def calcAngl(xyz1,xyz2,xyz3,xyz4):
81            vect1 = [xyz1[0]-xyz2[0], xyz1[1]-xyz2[1], xyz1[2]-xyz2[2]]
82            vect2 = [xyz3[0]-xyz4[0], xyz3[1]-xyz4[1], xyz3[2]-xyz4[2]]
83            vect3 = [vect1[0]-vect2[0], vect1[1]-vect2[1], vect1[2]-vect2[2]]
84            r1 = (vect1[0]**2 + vect1[1]**2 + vect1[2]**2)**0.5
85            r2 = (vect2[0]**2 + vect2[1]**2 + vect2[2]**2)**0.5
86            r3 = (vect3[0]**2 + vect3[1]**2 + vect3[2]**2)**0.5
87            angl = math.acos((r1**2 + r2**2 - r3**2)/(2*r1*r2))
88            angl = (angl/math.pi)*180
89            if angl>90:
90              angl = 180-angl
91            return angl
92
93          def avgCoords(atom1,atom2):
94            i = 0
95            cart = []
96            coords2 = atom2.GetParent().GetCoords(atom2)
97            for coord in atom1.GetParent().GetCoords(atom1):
98              cart.append((coord + coords2[i])/2)
99              i+=1
100           return cart
101
102         def GetNbrs(atom):
103           AtmList = []
104           for bond in atom.GetBonds():
105             AtmList.append(bond.GetNbr(atom))
106           return AtmList
107
108         def splitmol(mol,bond,otheratom,otheratom2):
109           newmol1 = OECreateOEMol(mol)  #Initiate new molecule
```

```
110          newmol2 = OECreateOEMol(mol)  #Initiate new molecule
111
112          atom1 = newmol1.GetAtom(HasAtomIdx(bond.GetBgnIdx()))
113          atom2 = newmol1.GetAtom(HasAtomIdx(bond.GetEndIdx()))
114          atom1.GetParent().SetCoords(atom1,avgCoords(atom1,otheratom))
115          atom1.SetName("A1")
116
117          atomlist = [atom2.GetIdx()]
118          newatoms = atomlist
119          forbidden = [atom1.GetIdx()]
120
121          while len(newatoms):    #recursively remove neighboring
122  atoms/bonds from molecule
123            for atom in newatoms:
124              newatoms.remove(atom)
125              Atom = newmol1.GetAtom(HasAtomIdx(atom))
126              if Atom is None: continue
127
128              AtmList = GetNbrs(Atom)
129              for nbor in AtmList:
130                if nbor.GetIdx() in forbidden:
131                  Atom2 = newmol1.GetAtom(HasAtomIdx(forbidden[0]))
132                  newmol1.DeleteBond(newmol1.GetBond(Atom,Atom2))
133                  continue
134                newatoms.insert(0,nbor.GetIdx())
135
136              for rmbond in Atom.GetBonds():
137                newmol1.DeleteBond(rmbond)
138
139              newmol1.DeleteAtom(Atom)
140
141          mollist = [newmol1]
142
143          atom1 = newmol2.GetAtom(HasAtomIdx(bond.GetEndIdx()))
144          atom2 = newmol2.GetAtom(HasAtomIdx(bond.GetBgnIdx()))
145
146          atom1.GetParent().SetCoords(atom1,avgCoords(atom1,otheratom2))
147          atom1.SetName("A1")
```

```
148
149          atomlist = [atom2.GetIdx()]
150          newatoms = atomlist
151          forbidden = [atom1.GetIdx()]
152
153          while len(newatoms):    #recursively remove neighboring
154   atoms/bonds from molecule
155            for atom in newatoms:
156              newatoms.remove(atom)
157              Atom = newmol2.GetAtom(HasAtomIdx(atom))
158              if Atom is None: continue
159
160              AtmList = GetNbrs(Atom)
161              for nbor in AtmList:
162                if nbor.GetIdx() in forbidden:
163                  Atom2 = newmol2.GetAtom(HasAtomIdx(forbidden[0]))
164                  newmol2.DeleteBond(newmol2.GetBond(Atom,Atom2))
165                  continue
166                newatoms.insert(0,nbor.GetIdx())
167
168              for rmbond in Atom.GetBonds():
169                newmol2.DeleteBond(rmbond)
170
171              newmol2.DeleteAtom(Atom)
172
173          mollist.append(newmol2)
174          return mollist
175
176        def linkmol(mol,bondorder):
177          atomlist = []
178          for atom in mol.GetAtoms():
179            if atom.GetName() == "A1":
180              atomlist.append(atom)
181              if len(atomlist) == 2:
182                mol.NewBond(atomlist[0],atomlist[1],bondorder)
183          return mol
184
185        if len(sys.argv)==3:
```

```
186           output = sys.argv[1]
187           if (output != '-osdf') and (output != '-osmi'):
188             print"usage: breed.py [-osdf | -osmi] file.sdf"
189             sys.exit(1)
190         else:
191             print"usage: breed.py [-osdf | -osmi] file.sdf"
192             sys.exit(1)
193
194         ifs = oemolistream()
195         ofs = oemolostream()
196         ofs.SetFormat(OEFormat_SDF)
197                                 # Read SD file
198         mollist = []
199         smilist = {}
200         if (ifs.open(sys.argv[2]) ==1):
201           for mol in ifs.GetOEMols():
202             OESuppressHydrogens(mol)
203             newmol = OECreateOEMol(mol)
204             smi = OECreateCanSmiString(mol)
205             smilist[smi] = 1
206             mollist.append(newmol)
207         else:
208           print"Error accessing SD file"
209           sys.exit(0)
210                                 # Loop over all bond pairs in all molecule
211   pairs.
212         i=0
213         for mol1 in mollist:
214           i+=1
215           for j in range(i,len(mollist)):
216             mol2 = mollist[j]
217             for bond1 in mol1.GetBonds():
218               if bond1.IsInRing():
219                 continue
220               else:
221                 for bond2 in mol2.GetBonds():
222                   if bond2.IsInRing():
223                     continue
```

```
                match_sense = bondmatch(bond1,bond2)  #is bond a match b/w
2 molecules?
                if(match_sense):
                  molname = mollist[i-1].GetTitle() + "_" +
mol2.GetTitle()

                  if match_sense==1 :
                    atom1 = bond2.GetBgn()
                    atom2 = bond2.GetEnd()
                  else:
                    atom1 = bond2.GetEnd()
                    atom2 = bond2.GetBgn()

                  fragments = []
                  fragments.append(splitmol(mollist[i-
1],bond1,atom1,atom2)) #split molecule 1

                  if match_sense==1:
                    atom1 = bond1.GetBgn()
                    atom2 = bond1.GetEnd()
                  else:
                    atom1 = bond1.GetEnd()
                    atom2 = bond1.GetBgn()

                  fragments.append(splitmol(mol2,bond2,atom1,atom2))
#split molecule 2
                  if match_sense == 1:
                    OEAddMols(fragments[0][0],fragments[1][1])       #2nd
half mol 2
                    bondorder = bond1.GetOrder()
                    newmol1 = linkmol(fragments[0][0],bondorder)
                    OEAddMols(fragments[1][0],fragments[0][1])       #2nd
half mol 1
                    newmol2 = linkmol(fragments[1][0],bondorder)
                  elif match_sense == 2:
                    OEAddMols(fragments[0][0],fragments[1][0])       #1st
half mol 2
                    bondorder = bond2.GetOrder()
```

```
                    newmol1 = linkmol(fragments[0][0],bondorder)
                    OEAddMols(fragments[0][1],fragments[1][1])        #2nd
half mol 2
                    newmol2 = linkmol(fragments[0][1],bondorder)

                  smi = OECreateCanSmiString(newmol1)
                  if not smilist.has_key(smi):
                    smilist[smi] = 1
                    if (output == '-osdf'):
                      OEWriteMolecule(ofs,newmol1)
                    else:
                      print smi,molname

                  smi = OECreateCanSmiString(newmol2)
                  if not smilist.has_key(smi):
                    smilist[smi] = 1
                    if (output == '-osdf'):
                      OEWriteMolecule(ofs,newmol2)
                    else:
                      print smi,molname
```